POLITECNICO
MILANO 1863

Design Document

version 1.0

11th December 2016

Authors:
Davide Moneta (808686)
Federico Oldani (806337)
Luca Oppedisano (878301)

Index

1 Introduction

1.1 Purpose

This document presents at different levels of detail the architecture of the system that ultimately will be implemented.
The document intends to describe the architectural decisions taken in the design process and justify them, also serving as an input for the next phase of the development process of the system.

1.2 Scope

We will design a software to manage the functionalities of PowerEnJoy: a new car sharing service. PowerEnJoy offers exclusively electric cars that allow the users to move around respecting the environment.
Using this platform customers are able to find and reserve one of our cars within a certain distance from their current location or a specified address. When they reach the chosen car, they can unlock it and drive it.
Users are charged with a per-minute fee, but the company incentivizes virtuous behaviors by offering a discount at the end of a trip if certain conditions are met.

1.3 Definitions, Acronyms, Abbreviations

Visitor: a generic person who access the website or the app without being authenticated
User: equivalent to registered user, a visitor who already performed the registration and has now access to the system
Car: with this term we always refer to one of our cars
Car tag: a term to label the status of a car, it can be "available" or "reserved" or "in use"
Available car: a car that is currently not in use nor reserved by anyone and has at least 30% of battery level
Reservation: it can be requested by a user for an available car, this gives the user the possibility to unlock the reserved car within a time span of 1 hour from the reservation time, after this time the reservation expires
Reserved car: a car that can be unlocked only by the one who reserved it
Ignition code: this code is sent to the user when a reservation is successful, user will have to type this code on the OBS lock screen in order to ignite the engine
Locking a car: after the passengers are all out of the car and the door are all closed the system counts 40 seconds and then locks the car. There are two types of locking: if the user wants to keep the car on pause, the car is locked and it remains on pause for that

user, otherwise the car is locked and made available to other users. System locks the car only if the car is inside the Safe Area
Pause: when the user turns off the engine the systems asks him on the OBS if he wants to put the car on pause, if the user taps "YES" the pause state is applied: the user keeps paying the per-minute fee until he unlocks it again
Ride or Trip: it starts when the engine is ignited or 2 minutes after the car is unlocked, and ends when the car is locked
Temporary bill: keeps track of the duration of the trip or the pause to calculate the amount of money to pay
Safe area: we consider a safe area the set of every possible legal parking spot within The Area, so all the places where it is possible to park the car without being fined
SPA: these are Special Parking Areas where cars can be parked and also recharged through a special plug
The System: the personification of the software, which manages all the functionalities of PowerEnJoy
MSM: Money Saving Mode, this can be enabled in the car after selecting a destination, this ensures to find parking solution to get a discount, optimizing the cost of the trip; the solution takes also into account the position of cars in The Area to ensure a uniform distribution
OBS: On Board Screen, it's the display inside the car, where the user can visualize informations about the car and the trip

1.4 Reference document

- RASD - PowerEnJoy v1.1
- Assignment AA 2016-2017 Software Engineering 2 - Project goal, schedule, and rules
- Lecture slides: Design Part I and Part II
- Sample Design Deliverable MyTaxyService

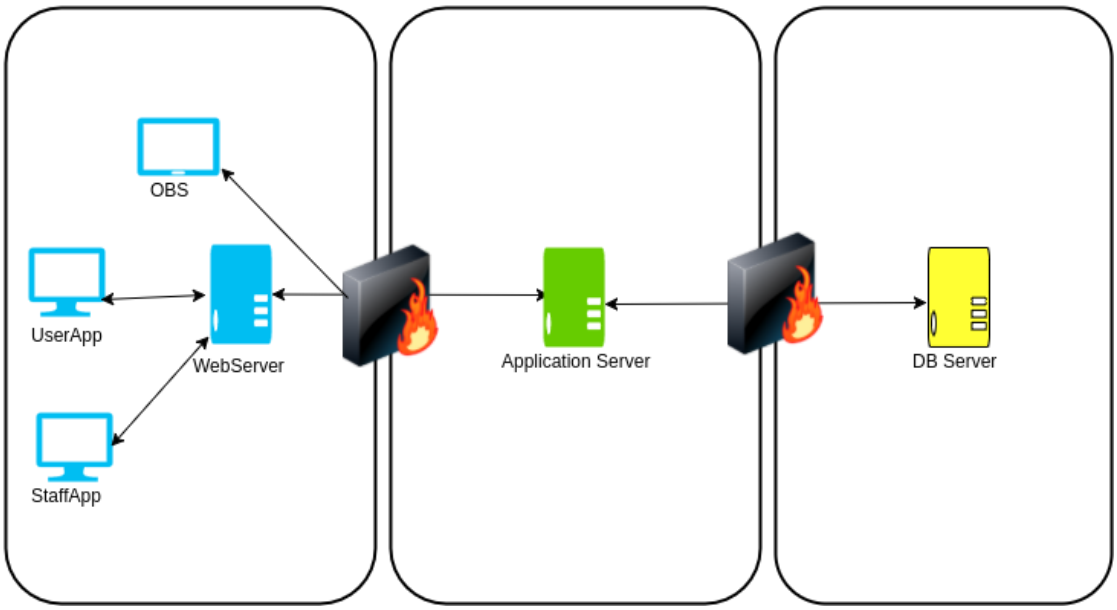
1.5 Document structure

- **Introduction**: this part describes the content of the document and its objectives.
- **Architectural design**:
    - **Overview**: it's the highest level view of our architecture, it shows how the tiers are linked and it explains what's the job of each one
    - **Components view**: this section will focus on the logical division of the components, each component will have its limited responsibility.
    - **Deployment**: This section shows how the software will be deployed to ensure proper functioning.

    - **Runtime view**: This section shows sequence diagrams that describe components interaction during task execution
    - **Component interfaces**: here we show the interfaces that permit the dialog between our components
    - **Selected architectural styles and patterns**: we present the architectural design patterns used
    - **Other design decisions**
- **Algorithm design**: We explain the characterizing logics implemented in our software
- **User interface design**: we present UX (user experience) and BCE (boundary control entity)
- **Requirement traceability**: we specify the components involved in the fulfillment of the goals presented in the RASD

2 Architectural design

2.1 Design overview



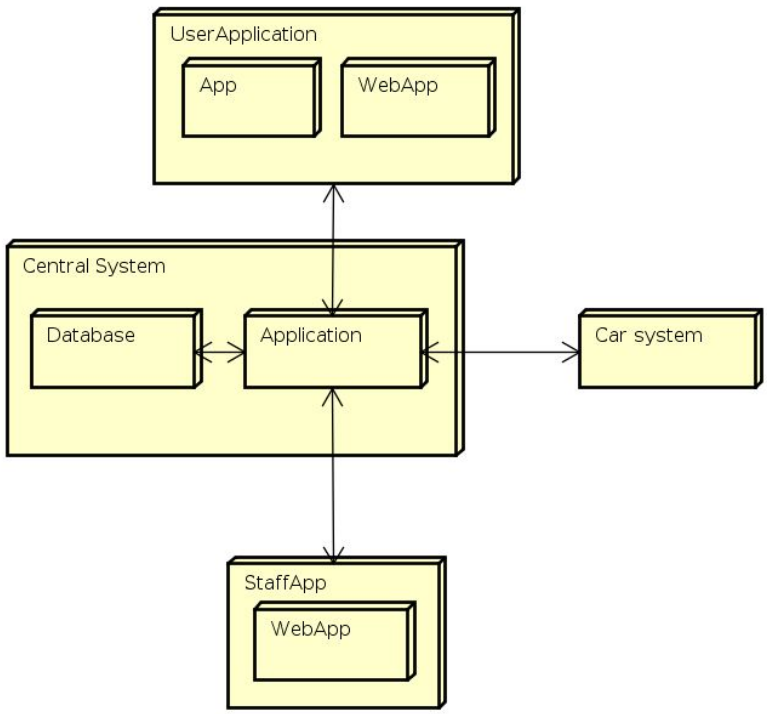
PowerEnJoy system will use a three tier architecture:

- The Presentation layer provides the interface through the client application and the on board system and loads contents of the web pages from the web server. Web server and Client device communicate through HTTP protocol.
- The Business layer contains the application server which offers the business logic and provide dynamic content. Web server, on board system and application server communicate through RESTful API in order to fulfill clients' requests.

- The Data layer contains a MySQL DBMS which interacts with the database to get and update information about clients, cars and SPAs.

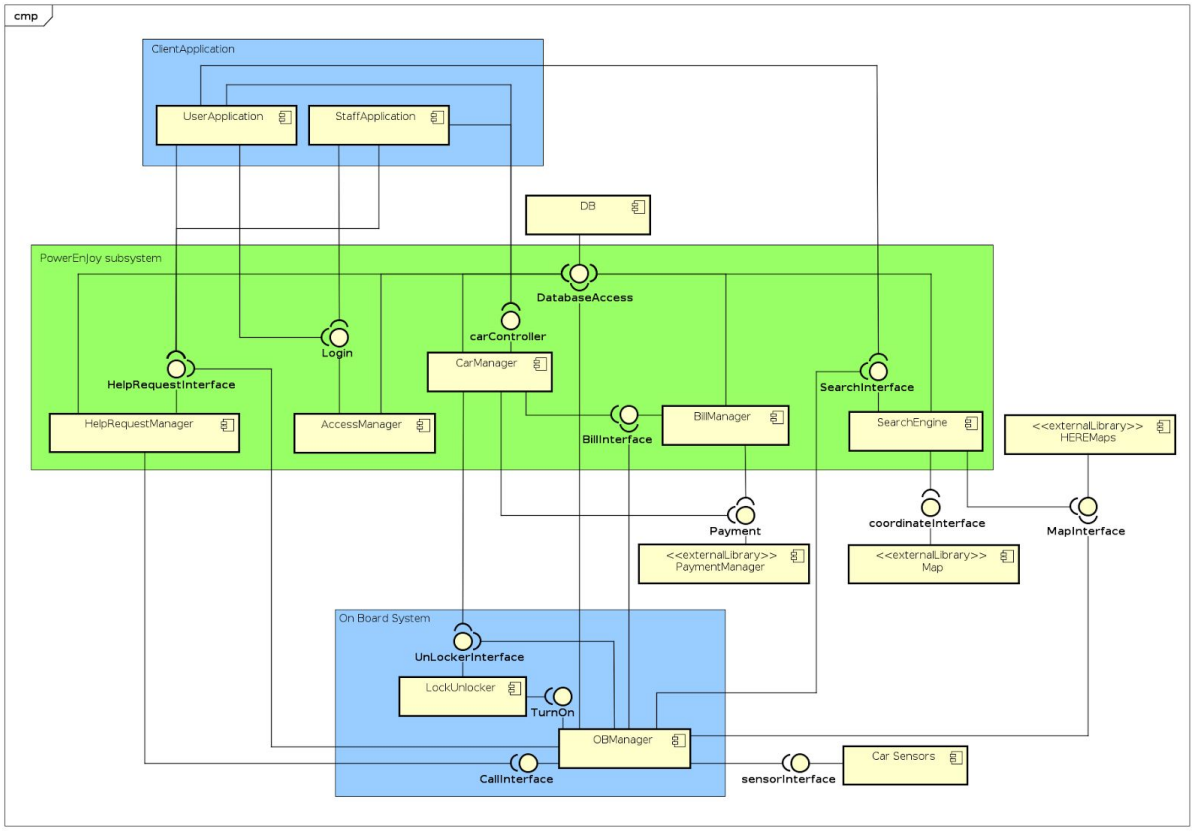
2.2 High level components and their interaction

PowerEnJoy system is composed by four main components. The core of the software is the Central System, which is a singleton, always operative to provide the business logic. Users and Staff can request services such as reservation, car unlocking and informations to the central system. This request is made in a synchronous way: the client (user or staff) will wait for an answer from the central system. Each car has an on board computer which runs our software that will interface with sensors and will communicate with the central system through an internet connection.



2.3 Component view

We used a Top-Down approach, meaning that we first thought about the goals we wanted to reach and the functionalities we wanted to provide, then we introduced the required component. We will divide our system in several components, each of them can be developed separately from the others and it can be easily substituted.
Some of the interfaces shown here are abstract interfaces and will have one concrete subinterface for each component that uses them, you can find more details on the concrete implementation in *Paragraph § 2.6* of this document.



2.3.1 Components' responsibility

Search Engine: uses the external HERE Maps component to find the coordinates of an address chosen by the user, if the user decided to use his current position this operation is not needed. This component also queries the database to find the available cars near the specified location, with a radius specified by the user. Using the HERE Maps component and marking the coordinates taken from the query it can build the map that will be shown to the client.

Map: a component developed by a third party company, which is able to check if a given position is in a SPA or in a Safe Area.

LockUnlocker: has the control over the mechanical locking system of the doors. It reduces the load on the OB Manager by putting it in sleeping mode when the car is locked. It also wakes up the OB Manager as soon as the doors are unlocked.

Database: contains Data about users, cars, reservations and the bills history.

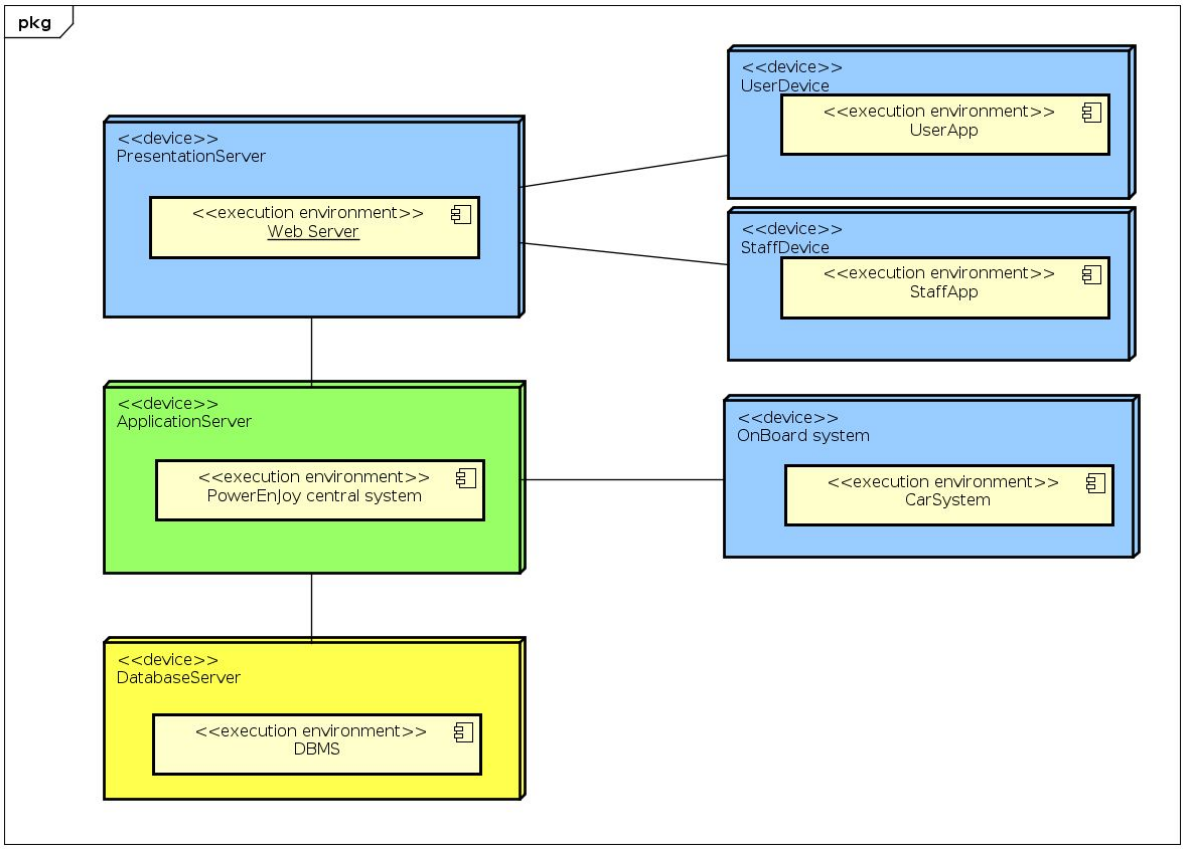
Car Manager: can access and update Data about cars in the Database and it sends the unlock command to the LockUnlocker. After setting a car as "in use" it creates an instance of Bill Manager for that car, and it locally keeps track of its position until the Bill Manager requests the payment for the trip. The link to the Payment Manager is needed in case of an expired reservation that requires the payment of a penalty fee. This component is called when a user confirms a reservation.

Help Request Manager: receives the requests from the users and manages them in a queue, as soon as a Staff member is available it forwards to him the first request in the queue.
Bill Manager: it's an object created for each car as soon as it's unlocked. After being notified by the OB manager that a trip has ended it will interact with the Payment Manager to carry out a transaction and it will update the Database to change the status of the car and to add the new data about the trip and the bill.
Car Sensors: this component provides data coming from car's sensor like weight and doors locking mechanism.
Payment Manager: is the external component that takes care of making the actual money transfer at the end of each trip.
Access Manager: has access to the credentials Table in the Database, is used to register new account and to make user sign in.
OB Manager: is the core of the system inside the car, it controls the sensors, the interface to communicate with the driver (OBS) and he has network access to communicate with the Central System. It's notified when the engine is turned on, so it can tell to the Bill Manager to start the trip, and when the engine is turned off so that it can start the timers and begin monitoring seats and doors sensors before asking to the LockUnlocker to lock the car. After the unlock it tells to the Bill Manager to proceed with the payment. Access to the SPA table in the Database is required in order to provide the MSM and the Navigate to SPA functionalities.
HERE Maps: is the external mapping service used to provide turn by turn navigation and maps.

Further notes about the components behavior:

- It would be possible to have a "live" map with the real time position of the cars in use but this would require a lot of requests towards the Car Manager so we avoided the implementation of this feature. The Staff Application is implemented so that a staff member is able to retrieve the position of a single car if necessary.
- The control for safe/unsafe/SPA area is not made in real-time, but only when the engine is turned off.
- Help requests behave differently based on the sender's platform: when sent from the OBS it will have the car's position attached and if accepted a VoIP call will start inside the car, if sent from the web or mobile application no position will be attached and if accepted a chat with a staff operator will be opened.
- Each Staff operator works with his own terminal in the office, each terminal runs an instance of Staff Application

2.4 Deployment

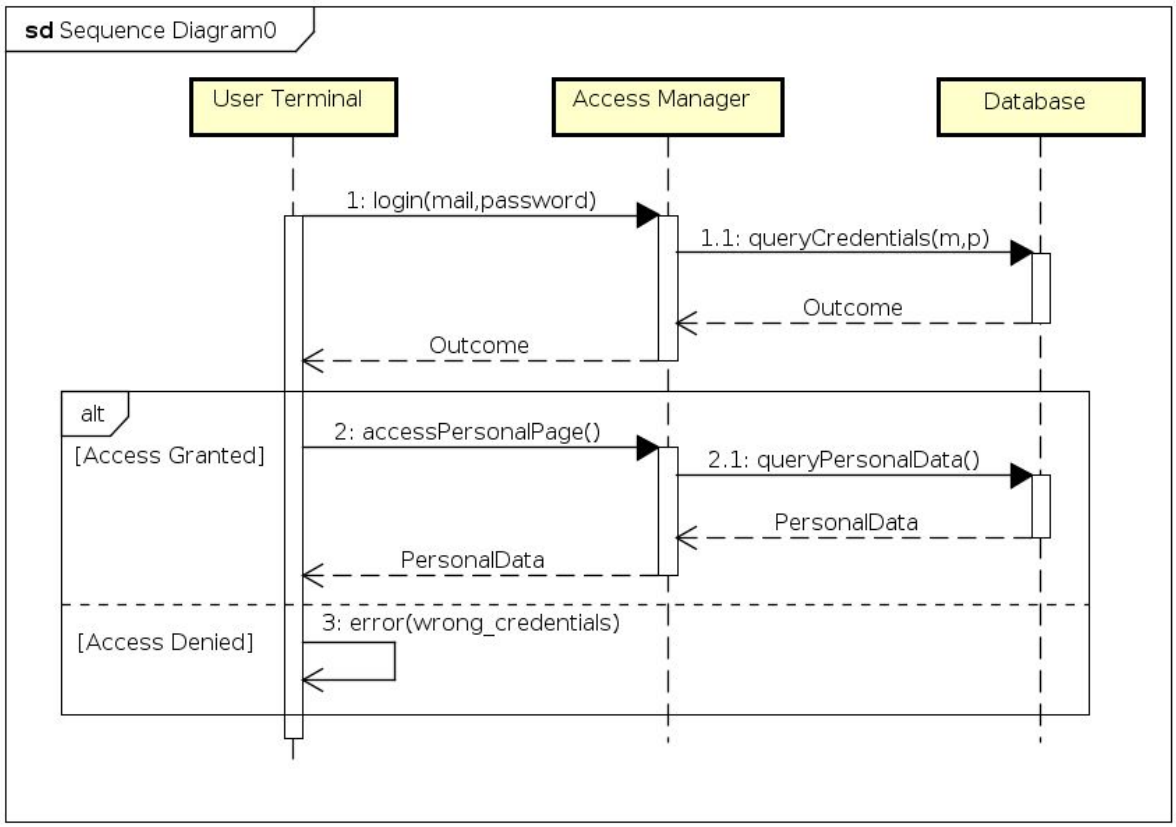


Each color represents a different tier described in *paragraph § 2.1*. Our software will be deployed like follows:

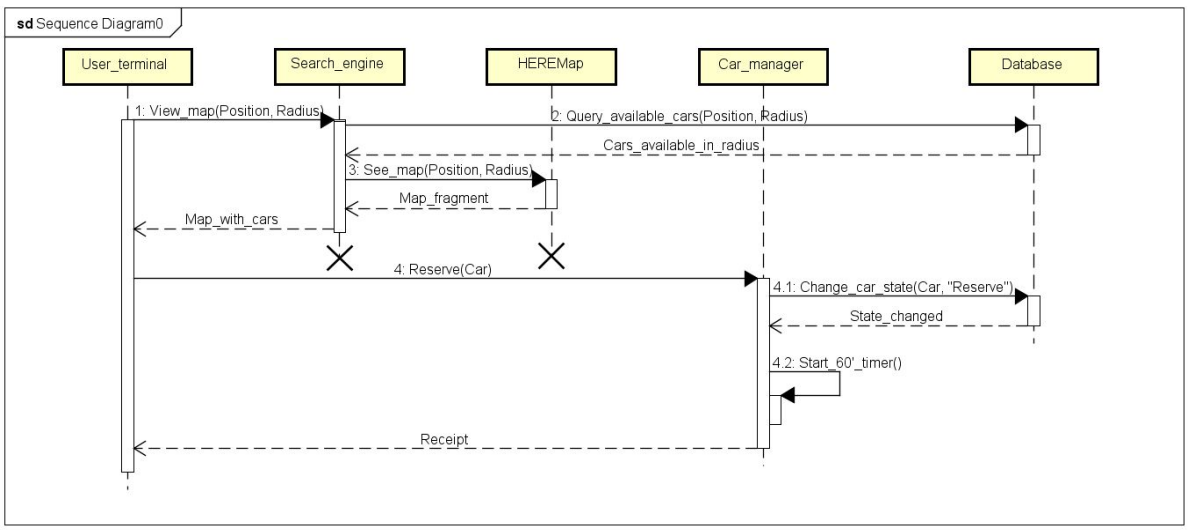
- User or Staff device will run our application
- Presentation Server will provide static content to the apps and will ask for dynamic content to the Application Server. PowerEnJoy will use an external server to host the website.
- Application Server: This server belongs to the Company, it contains our business logic and it can query and update the Database.
- Database Server: PowerEnJoy will use a cloud service to store information.

2.5 Runtime view

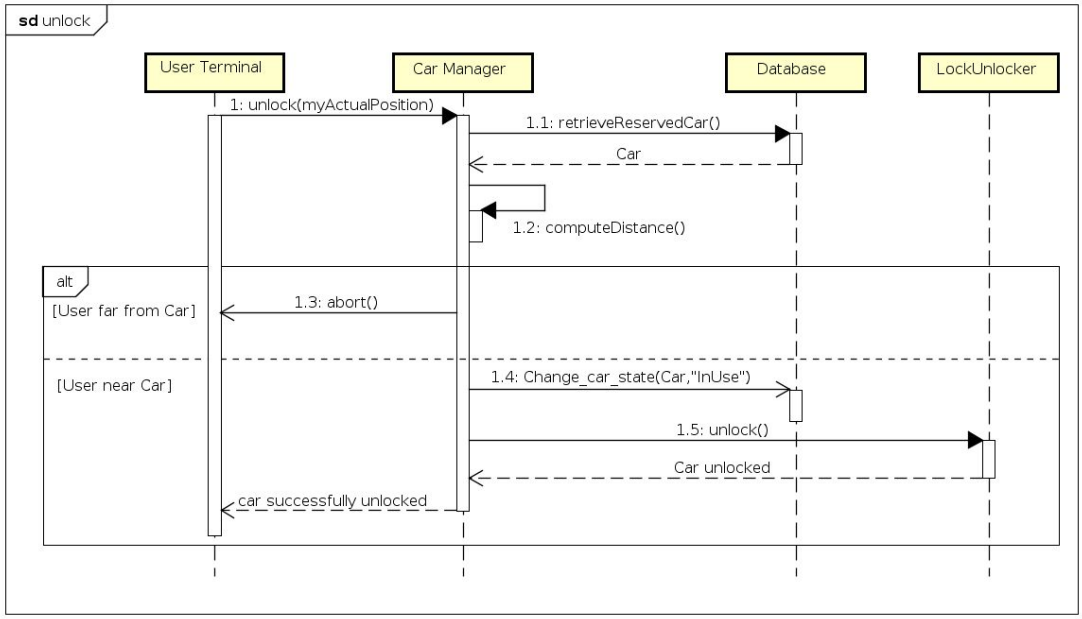
2.5.1 Login



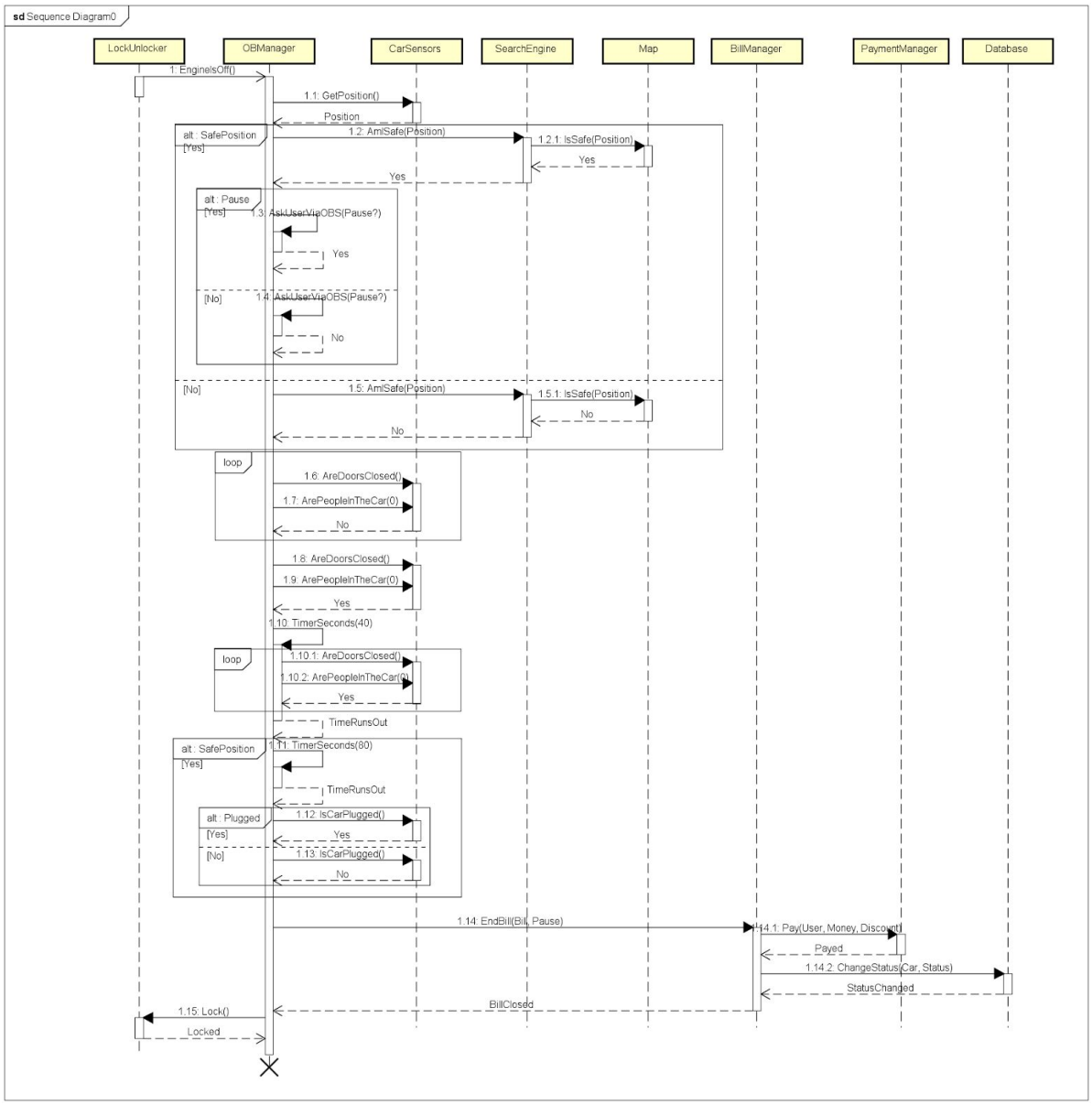
2.5.2 Search and reservation



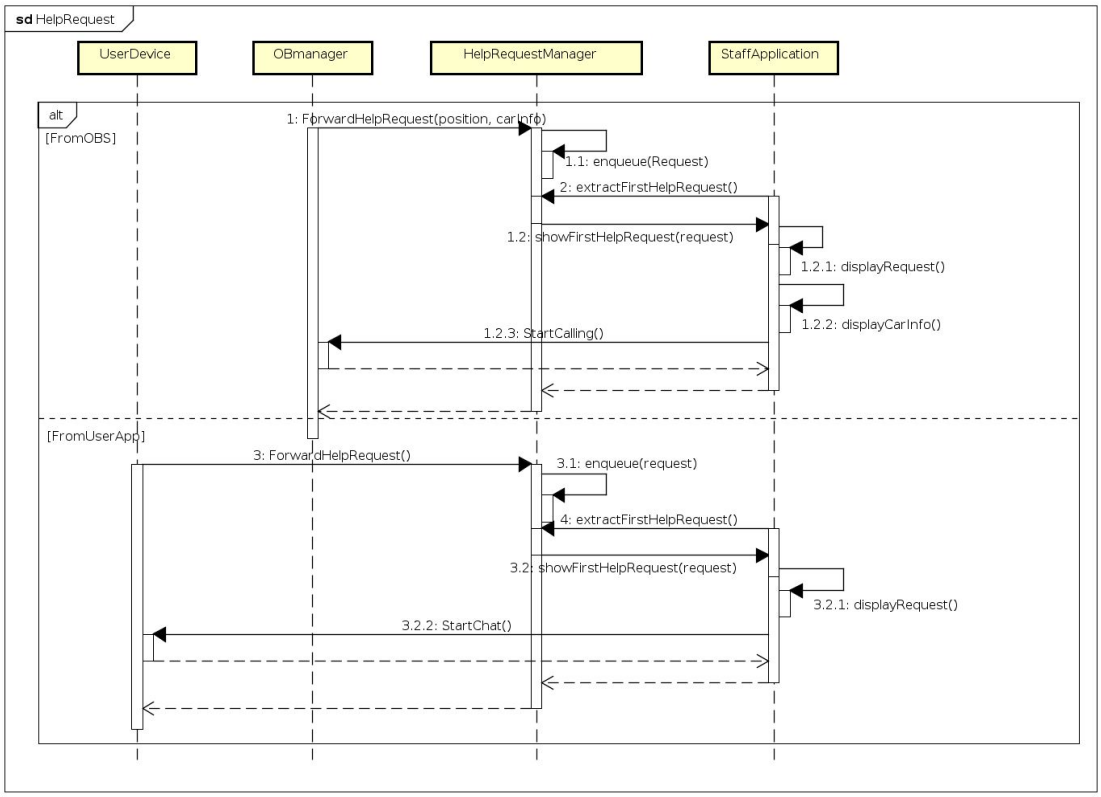
2.5.3 Unlock


sd unlock
User Terminal
Car Manager
Database
LockUnlocker
1: unlock(myActualPosition)
1.1: retrieveReservedCar()
Car
1.2: computeDistance()
alt
[User far from Car]
1.3: abort()
[User near Car]
1.4: Change_car_state(Car,"InUse")
1.5: unlock()
Car unlocked
car successfully unlocked

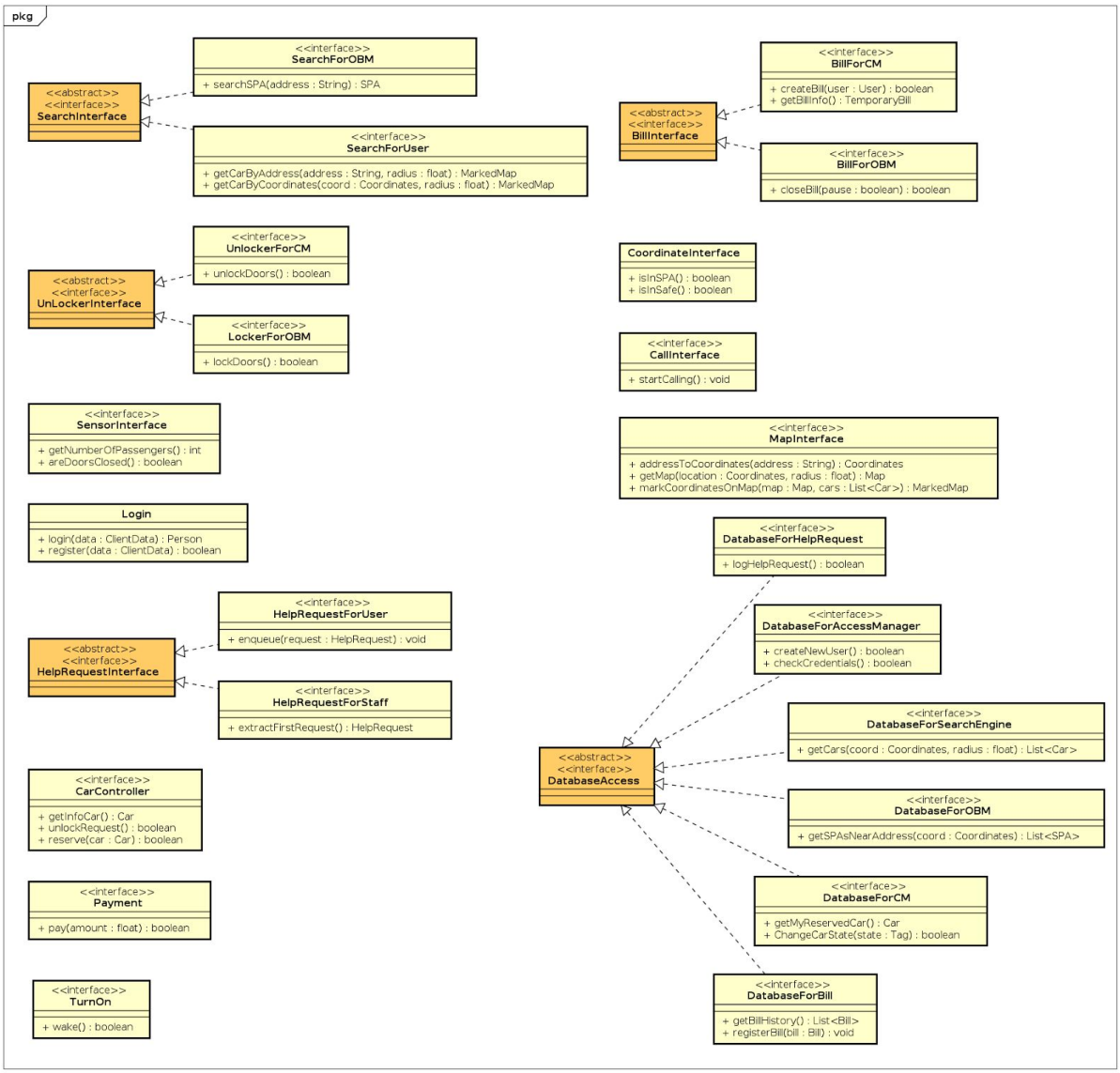
2.5.4 Lock and Pay



2.5.5 Staff help user



2.6 Component interfaces



For external library interface methods consult their official documentation.

2.7 Selected architectural styles and patterns

As written above, for the PowerEnJoy system we chose a three tier architecture:
An NGINX Web Server will manage the connections with the clients, provide static content and will ask for dynamic content and business logic to a Glassfish Application Server.
The on board system and web server will communicates with the application server through RESTful web services. JDBC protocol allow the application server to communicate with MySQL database.

Patterns that we will use:

- **MVC** is our main high level guideline: our three tiers correspond to the Model (Database), View (Presentation) and Controller (Business)
- **Client-Server:** we want to implement a thin client without any logic control. All decision and algorithm run in the Application server.

2.8 Other design considerations

The distribution of our system has to be analyzed under different points of view: we want to deploy different components into different devices, but we don't want to create fragmentation in the process, so each task is bound to a specific device.
In our architecture the internal structure of each component is independent from the others and each tier is protected by a firewall to prevent fraudulent access.
We decided to separate the Application Server and the Web Server so that the first can take care only of data retrieval and elaboration, while the other manages the actual interaction with the user.
We preferred NGINX instead of Apache as Web Server, so the number of concurrent clients' connection can scale with less worries on the server resources.

3 Algorithm design

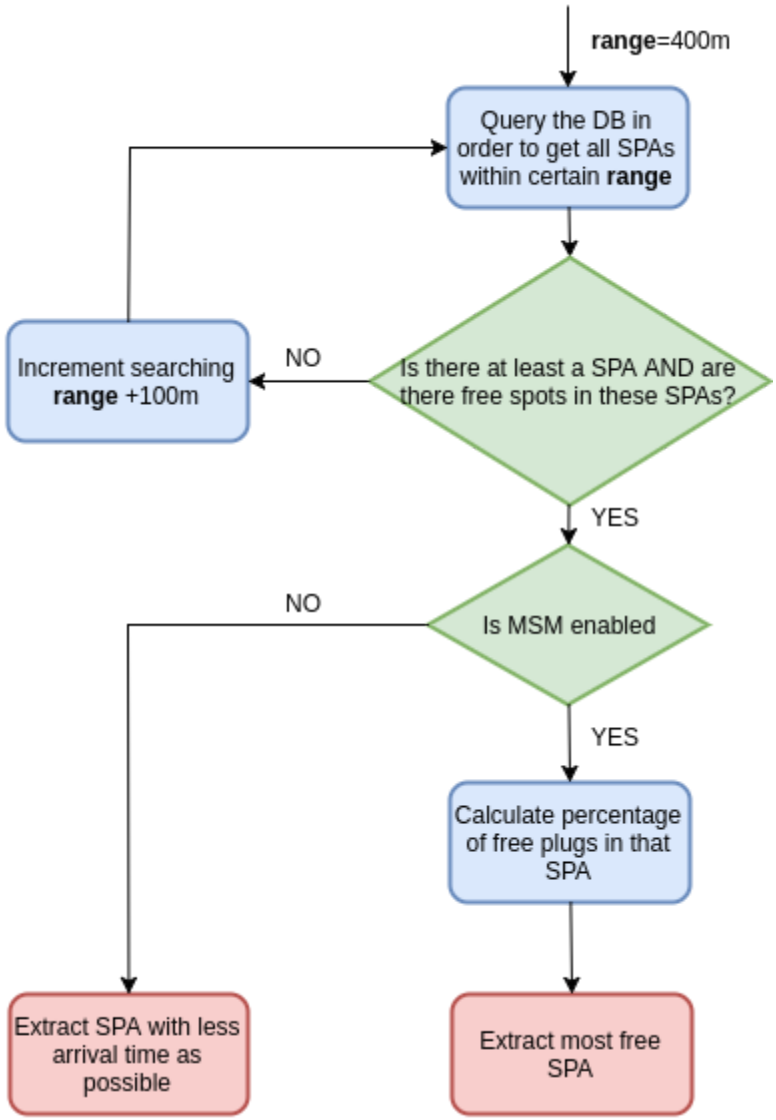
3.1 Bill computing

The Bill amount is computed as trip duration (in minutes) multiplied by per-minute fee, but a discount or a penalty may be applied, according to the following table:

Condition	Variation on Bill
numberOfPassengers ≥ 2	-10%
batteryChargeLeft ≥ 50%	-20%
pluggedInSPA == True	-30%
distanceFromNearestSPA > 3km OR batteryChargeLeft < 20%	+30%

3.2 SPA finding (with or without MSM)

The following flowchart explains how works the algorithm to find SPA. It begins querying the database in order to get all SPAs within a range of 400m from the given position.
If the user taps on *SearchNearestSPA* the system will use their position, otherwise a destination address given by them.
If MSM is enabled the algorithm selects the SPA with the highest percentage of plugs available among those within a selected area around the destination. Initially the selected area is a circle of 400m of radius around the destination, but if there isn't at least one SPA with free plugs the radius is incremented of 100m until the conditions are met.



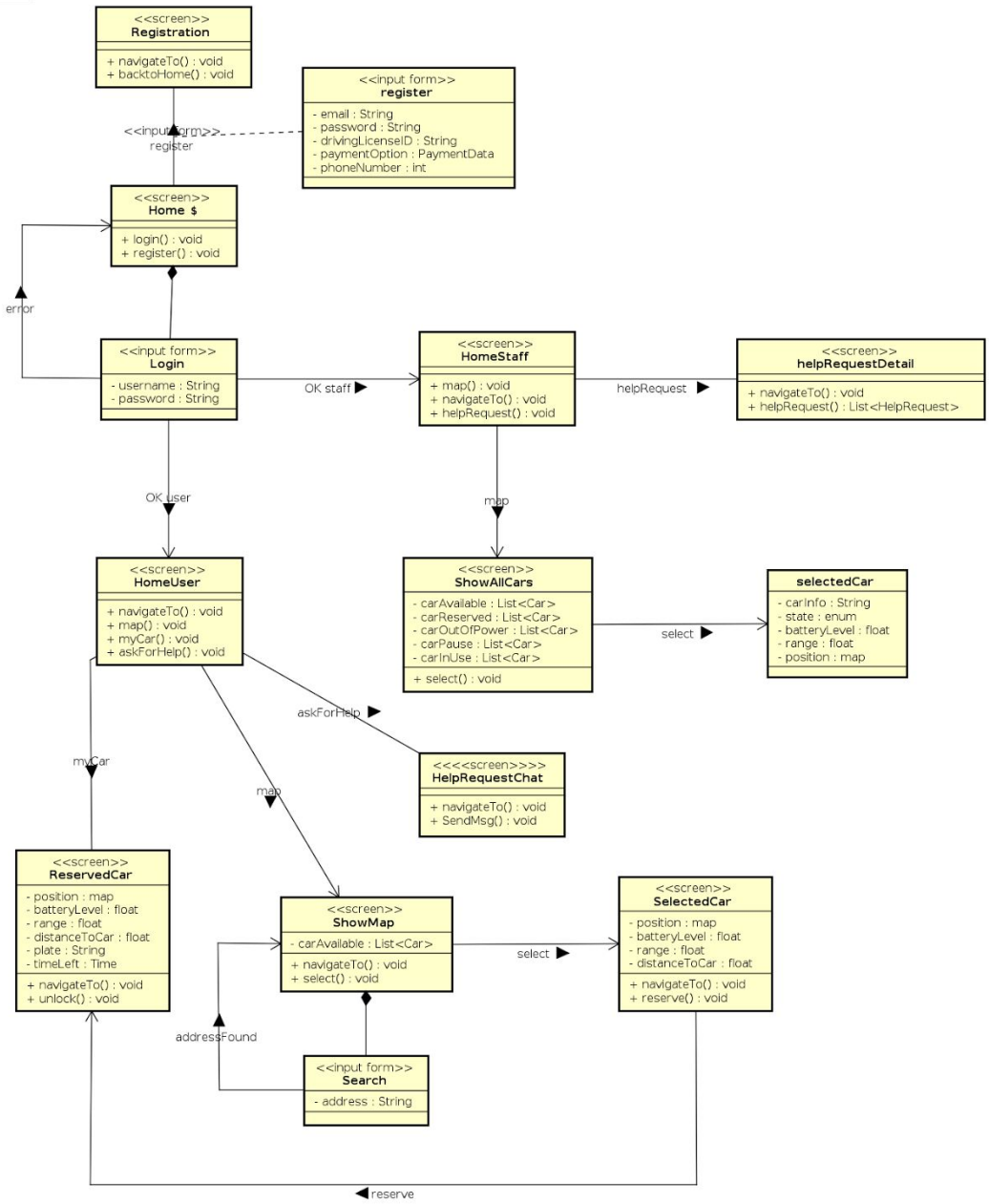
4 User interface design

4.1 UX Diagrams

Here we present the User Experience, each *square* of the following diagram corresponds to an application *screen*. Mockups of the user interface can be found in RASD document at Paragraph § 3.1.1.

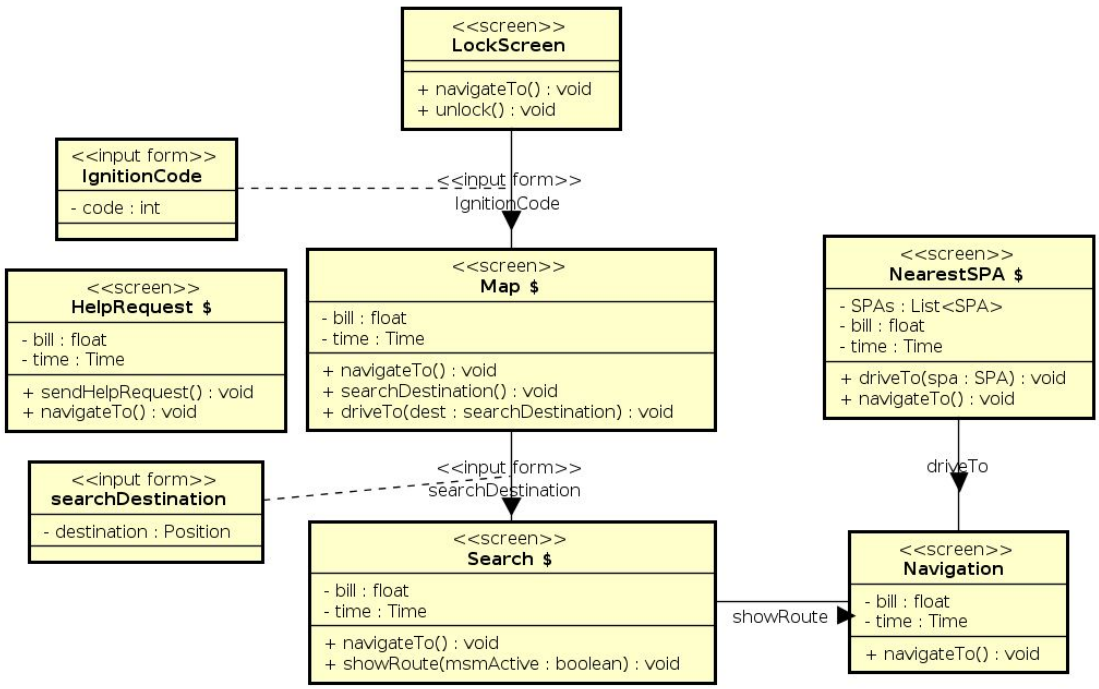
4.1.1 Application

Notes: registration screen is for users only, the staff is manually inserted in the database.



4.1.2 OBS

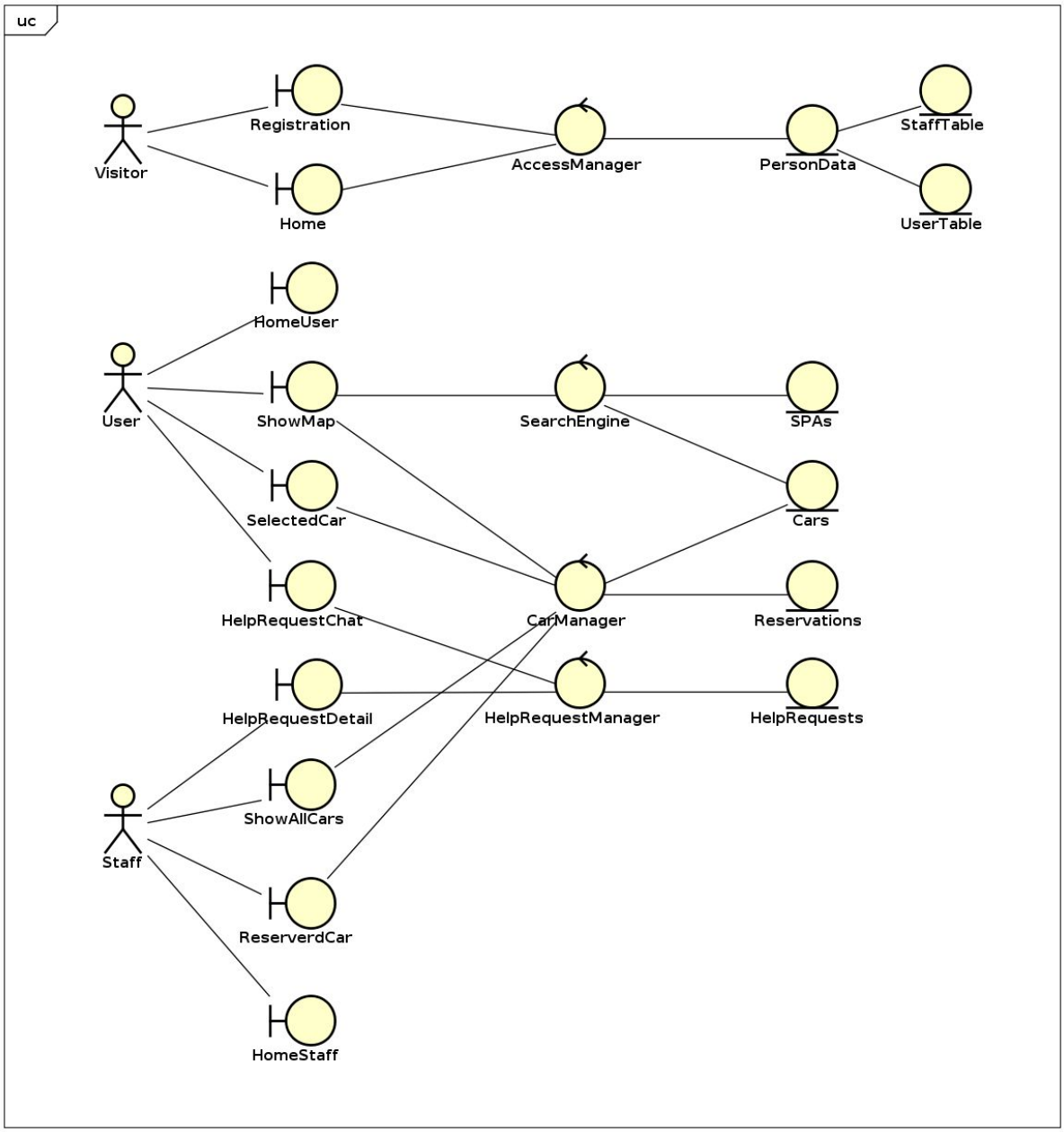
In the following diagram it's presented the User Experience of on board screen. Notes: the symbol $ means that the link for the screen is available from all other screens.



4.2 BCE Diagrams

4.2.1 Application

Notes: *External Library* and not directly linked components were omitted in the following diagram.



4.2.2 On Board System

On board system BCE is omitted because it's trivial: the OBManager is the controller for all the boundaries and works as proxy and interacts with other central system's components.

5 Requirement traceability

Here we list all the components involved in order to reach the goals of PowerEnJoy

[G1] Allow visitors having a valid driving license and a valid payment option to register to the system and so to become a user

- User Application
- Access Manager
- Database

[G2] Allow users to log into the system

- User Application
- Access Manager
- Database

[G3] Allow users to find a car within certain distance (chosen by the user) from a specified location or from their current location

- User Application
- Search Engine
- Database
- HERE Maps

[G4] Allow users to reserve a car up to an hour before picking it up

- User Application
- Car Manager
- Database

[G5] If the reserved car is not picked up, The System charges the user 1€ and the reservation expires

- Car Manager
- Payment Manager
- Database

[G6] Allow to unlock a car

- User Application
- Car Manager
- Database
- Lock-Unlocker

[G7] Allow users to drive the reserved car and to know the current bill based just on trip time, with the discount applied only at the end of the trip

- OBManager
- Bill Manager

[G8] Allow users to know where the nearest SPAs are located via OBS

- OBManager
- Search Engine
- Map
- HERE Maps

[G9] Allow users to enable MSM to reach their destination paying as little as possible

- OBManager
- Search Engine
- Database
- Map
- HERE Maps

[G10] Allow users to plug cars into SPAs' power grid and get the discount of 30% on the last ride

- OBManager
- Car Sensors
- Bill Manager
- Payment Manager
- Database

[G11] Lock the car after user parks and leaves the car

- Lock-Unlocker
- OBManager
- Car Sensors
- Search Engine
- Map
- Database

[G12] Allow users to park and pause the trip to keep the car reserved if they have to go for a stopover

- Lock-Unlocker
- OBManager
- Car Sensors
- Search Engine
- Map
- Database

[G13] Allow users to contact the support by a phone call, if they need assistance

- User Application
- Help Request Manager
- Staff Application

[G14] Allow staff to know where the PowerEnjoy cars are, their battery level and reservation history

- Staff Application
- Car Manager
- Database

[G15] Allow staff login to the system

- Staff Application
- Login Manager
- Database

[G16] If the system detects at least two passengers (driver excluded), the system applies a discount of 10% on the last ride

- OBManager
- Car Sensors
- Bill Manager
- Payment Manager

[G17] If the car is left with no more than 50% of the battery empty, the system applies a discount of 20% on the last ride

- OBManager
- Car Sensors
- Bill Manager
- Payment Manager

[G18] If car is left at more than 3 km from the nearest power grid station or with more than 80% the battery empty, the system charges 30% more on the last ride to compensate for the cost required to re-charge the car on-site

- OBManager
- Car Sensors
- Search Engine
- Map
- Bill Manager
- Payment Manager

6 Hours of work

6.1 Luca Oppedisano

18/11/2016	1h00’
21/11/2016	1h00’
22/11/2016	2h00’
23/11/2016	5h00’
27/11/2016	1h30’
28/11/2016	1h00’
29/11/2016	2h00’
30/11/2016	6h00’
1/12/2016	4h00’
3/12/2016	0h30’
6/12/2016	0h30’
7/12/2016	5h30’
8/12/2016	2h00’
9/12/2016	3h30’
10/12/2016	4h30’

6.2 Federico Oldani

18/11/2016	1h00’
21/11/2016	2h00’
22/11/2016	4h00’
23/11/2016	2h00’
25/11/2016	1h00’
28/11/2016	1h00’
29/11/2016	2h00’
30/11/2016	5h30
1/12/2016	2h00’
2/12/2016	4h30’
3/12/2016	1h30’
6/12/2016	3h30’
7/12/2016	5h00’
8/12/2016	1h00’
9/12/2016	2h30’
10/12/2016	3h00’

6.3 Davide Moneta

28/11/2016	1h00’
29/11/2016	2h30’
30/11/2016	4h45’
1/12/2016	3h00’
3/12/2016	2h30’
6/12/2016	0h30’
7/12/2016	3h30’
9/12/2016	1h00’
10/12/2016	3h30’